- **SD 卡接口的完整规范**

  容量：32MB/64MB/128MB/256MB/512MB/1GByte
  兼容规范版本 1.01
  卡上错误校正
  支持 CPRM
  两个可选的通信协议：SD 模式和 SPI 模式
  可变时钟频率 0－25MHz
  通信电压范围：2.0-3.6V 工作电压范围:2.0-3.6V
  低电压消耗：自动断电及自动睡醒，智能电源管理
  无需额外编程电压
  卡片带电插拔保护
  正向兼容 MMC 卡
  高速串行接口带随即存取
  －－－支持双通道闪存交叉存取
  －－－快写技术：一个低成本的方案，能够超高速闪存访问和高可靠数据存储
  －－－最大读写速率：10Mbyte/s
  最大 10 个堆叠的卡（20MHz,Vcc=2.7-3.6V)
  数据寿命：10 万次编程/擦除
  CE 和 FCC 认证
  PIP 封装技术
  尺寸：24mm 宽×32mm 长×1.44mm 厚

- 说明：

  本 SD 卡高度集成闪存，具备串行和随机存取能力。可以通过专用优化速度的串行接口访问，数据传输可靠。接口允许几个卡垛叠，通过他们的外部连接。接口完全符合最新的消费者标准，叫做 SD 卡系统标准，由 SD 卡系统规范定义。
  SD 卡系统是一个新的大容量存储系统，基于半导体技术的变革。
  它的出现，提供了一个便宜的、结实的卡片式的存储媒介，为了消费多媒体应用。
  SD 卡可以设计出便宜的播放器和驱动器而没有可移动的部分。
  一个低耗电和广供电电压的可以满足移动电话、电池应用比如音乐播放器、个人管理器、掌上电脑、电子书、电子百科全书、电子词典等等。
  使用非常有效的数据压缩比如 MPEG，SD 卡可以提供足够的容量来应付多媒体数据。

- 框图：

  SD 卡上所有单元由内部时钟发生器提供时钟。接口驱动单元同步外部时钟的 DAT 和 CMD 信号到内部所用时钟。
  本卡由 6 线 SD 卡接口控制，包括：CMD,CLK,DAT0-DAT3。
  在多 SD 卡垛叠中为了标识 SD 卡，一个卡标识寄存器(CID)和一个相应地址寄存器（RCA）预先准备好。
  一个附加的寄存器包括不同类型操作参数。
  这个寄存器叫做 CSD。 使用 SD 卡线访问存储器还是寄存器的通信由 SD 卡标准定义。卡有自己的电源开通检测单元。 无需附加的主复位信号来在电源开启后安装卡。 它防短路，在带电插入或移出卡时。 无需外部编程电压。 编程电压卡内生成。 SD 卡支持第二接口工作模式 SPI。 如果接到复位命令（CMD0）时，CS 信号有效（低电平），SPI 模式启用。

- 接口

  该 SD 卡的接口可以支持两种操作模式：SD 卡模式、SPI 模式

主机系统可以选择以上其中任一模式，SD 卡模式允许 4 线的高速数据传输。 SPI 模式允许简单通用的 SPI 通道接口， 这种模式相对于 SD 模式的不足之处是丧失了速度。

SD 卡模式针脚定义

针脚	名称	类型	描述
1	CD DAT3	I/O/PP	卡监测数据位 3
2	CMD	PP	命令/回复
3	Vss	S	地
4	Vcc	S	供电电压
5	CLK	I	时钟
6	Css2	S	地
7	DAT0	I/O/PP	数据位 0
8	DAT1	I/O/PP	数据位 1
9	DAT2	I/O/PP	数据位 2

1：S：电源供电，I：输入 O：输出 I/O：双向 PP：I/O 使用推挽驱动

SD 卡的总线概念

SD 总线允许强大的 1 线到 4 线数据信号设置。当默认的上电后，SD 卡使用 DAT0。 初始化之后，主机可以改变线宽（译者按：即改为 2 根线，3 根线。。。）。混和的 SD 卡连接方式也适合于主机。在混和连接中 Vcc，Vss 和 CLK 的信号连接可以通用。但是，命令，回复，和数据（DAT0～3）这几根线，各个 SD 卡必须从主机分开。

这个特性使得硬件和系统上交替使用。SD 总线上通信的命令和数据比特流从一个起始位开始，以停止位中止。

CLK：每个时钟周期传输一个命令或数据位。频率可在 0～25MHz 之间变化。SD 卡的总线管理器可以不受任何限制的自由产生 0～25MHz 的频率。

CMD：命令从该 CMD 线上串行传输。一个命令是一次主机到从卡操作的开始。命令可以以单机寻址（寻址命令）或呼叫所有卡（广播命令）方式发送。

回复从该 CMD 线上串行传输。一个命令是对之前命令的回答。回复可以来自单机或所有卡。

DAT0～3：数据可以从卡传向主机或副 versa。数据通过数据线传输。

SPI 模式针脚定义

针脚	名称	类型	描述
1	CS	I	片选（负有效）
2	DI	I	数据输入
3	Vss	S	地
4	Vcc	S	供电电压
5	CLK	I	时钟
6	Vss2	S	地
7	DO	O	数据输出
8	RSV	--	
9	RSV	--	

1：S：电源供电，I：输入 O：输出 I/O：双向 PP：I/O 使用推挽驱动

注意：SPI 模式时，这些信号需要在主机端用 10~100K 欧的电阻上拉。

SPI 总线概念

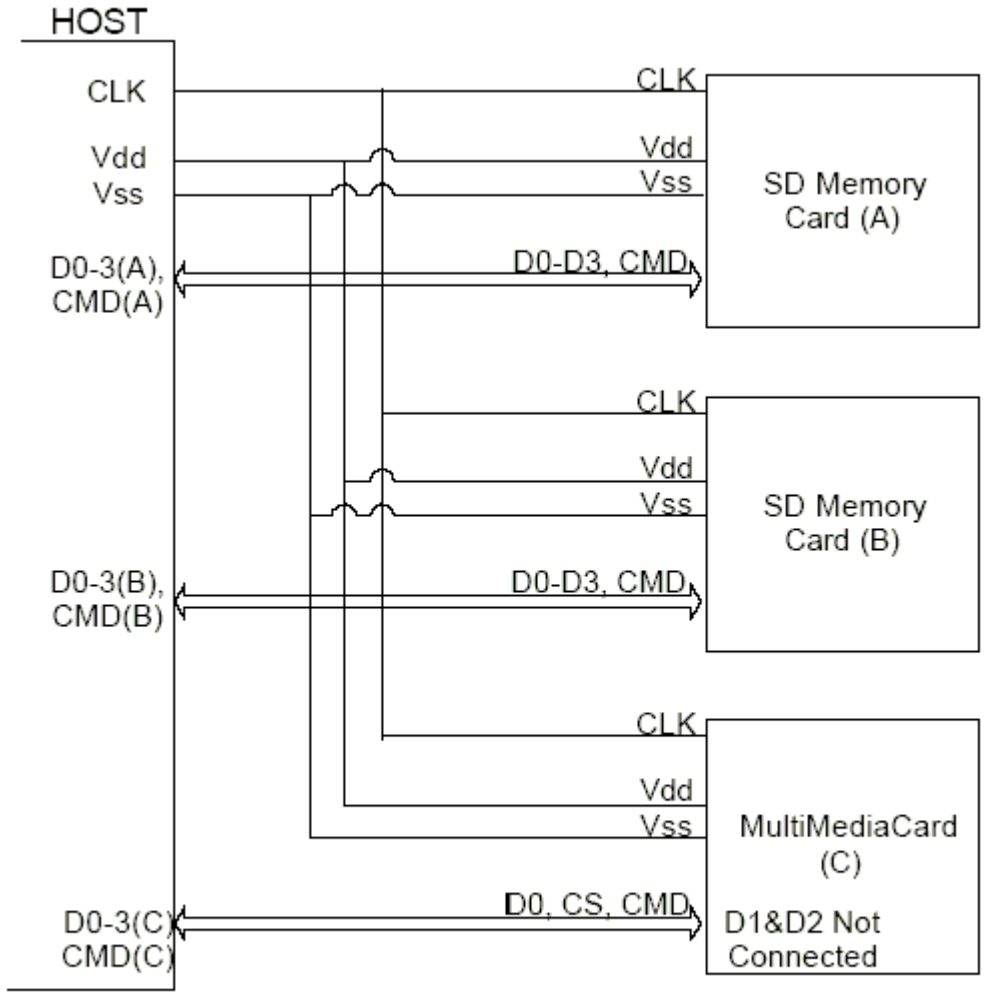
SPI 总线允许通过 2 通道（数据入和出）传输比特数据。SPI 兼容模式使得 MMC 主机系统通过很小的改动就可以使用ＳＤ卡。SPI 模式使用字节传输。所有的数据被融合到一些字节中并 aligned to the CS signal（可能是：同过 CS 信号来校正）。SPI 模式的优点就是简化主机的设计。特别的，MMC 主机需要小的改动。SPI 模式相对于 SD 模式的不足之处是丧失了速度性能。

直流特性，完全最大值，评估最大值评估指即使在瞬间也不能超出限制电压。当你在归定的最大值评估范围内使用该产品，不会出现永久性损坏。但是这并不能保证正常的逻辑操作。

SD 卡接口的完整规范（二）

回复从该 CMD 线上串行传输。一个命令是对之前命令的回答。回复可以来自单机或所有卡。

DAT0～3：数据可以从卡传向主机或副 versa。数据通过数据线传输。



SD Card bus Topology

SD 卡总线拓扑

SPI 模式针脚定义

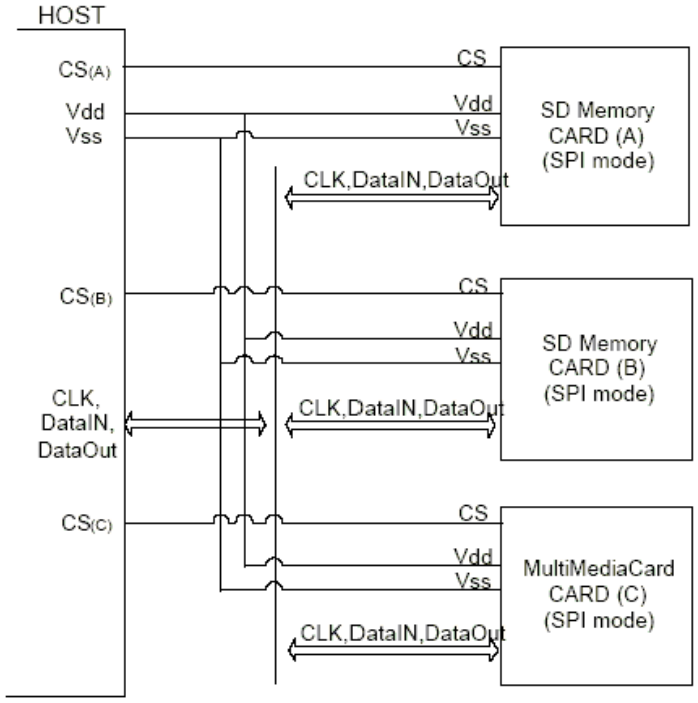
针脚	名称	类型	描述
1	CS	I	片选（负有效）
2	DI	I	数据输入
3	Vss	S	地
4	Vcc	S	供电电压
5	CLK	I	时钟
6	Vss2	S	地
7	DO	O	数据输出
8	RSV	--	
9	RSV	--	

1：S：电源供电，I：输入 O：输出 I/O：双向 PP：I/O 使用推挽驱动

注意：SPI 模式时，这些信号需要在主机端用 10~100K 欧的电阻上拉。

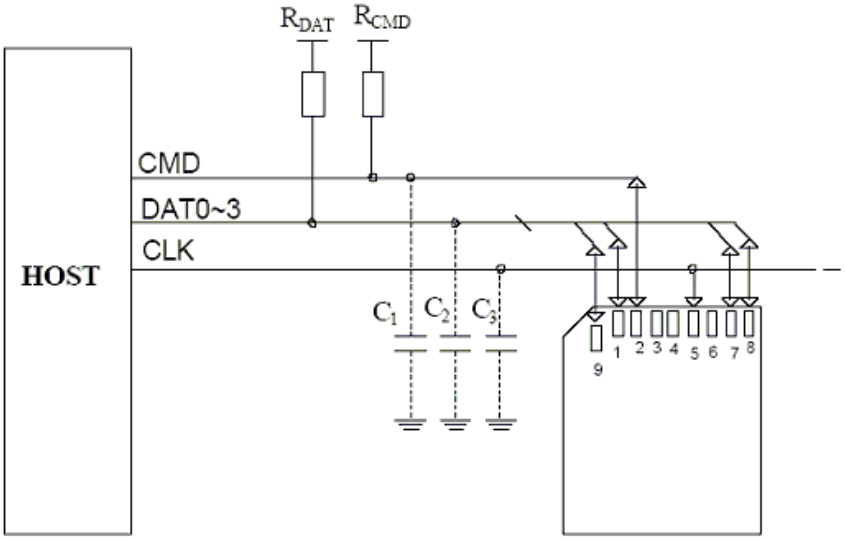
SPI 总线概念

SPI 总线允许通过 2 通道（数据入和出）传输比特数据。SPI 兼容模式使得 MMC 主机系统通过很小的改动就可以使用 S D 卡。SPI 模式使用字节传输。所有的数据被融合到一些字节中并 aligned to the CS signal（可能是：同过 CS 信号来校正）。SPI 模式的优点就是简化主机的设计。特别的，MMC 主机需要小的改动。SPI 模式相对于 SD 模式的不足之处是丧失了速度性能。

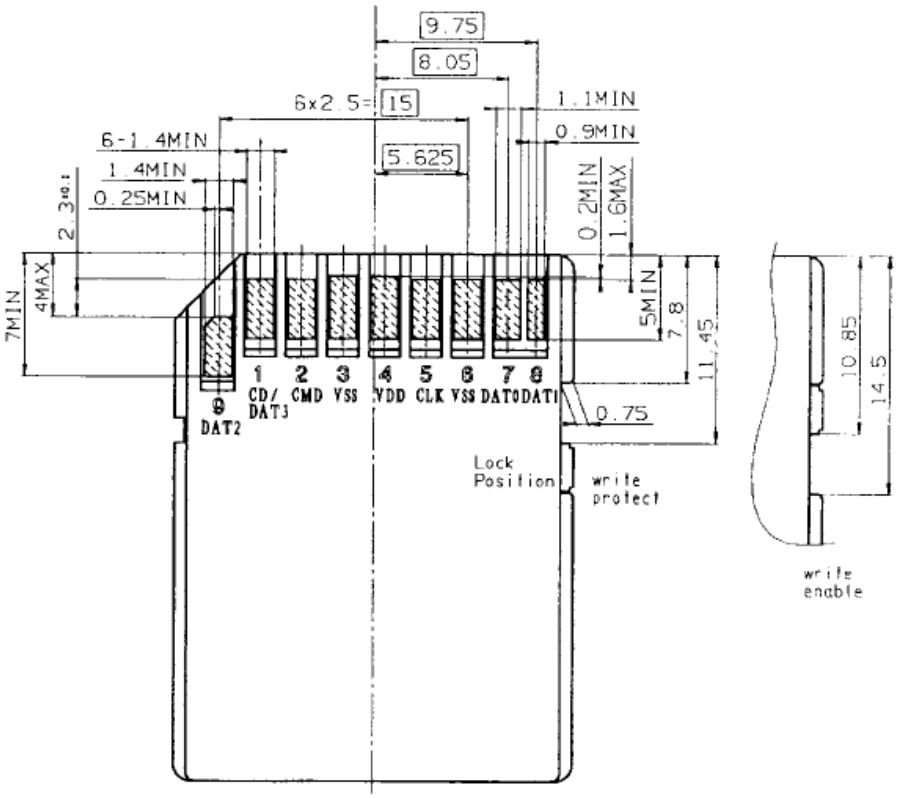


SPI mode bus topology

SD 卡的电特性



SD card Connection diagram



SD 卡的连接电路图

```
#include <iom32v.h>
#include <macros.h>
#include "1011.h"
#define uchar unsigned char
#define uint unsigned int
#define MMC_CS_PIN            BIT(4)         //PORTB.4
#define MMC_PORT              PORTB

uchar reading=0,a=0,pointer=0;
void sd_port_init()
{
MMC_PORT =MMC_CS_PIN;
}
uchar BUFFER[512];                          //扇区缓冲区
uint i=0;
void delay_nus(uint n)
{
unsigned char      b;
     for (b = 1; b<n; b++)
       ;
}
//*****************************************************************************
//Send a Command to MMC/SD-Card
//Return: the second byte of response register of MMC/SD-Card
//*****************************************************************************
uchar SD_Write_Command(uchar cmd,unsigned long arg)
{
       uchar tmp;
       uchar retry=0;

       //MMC_PORT =MMC_CS_PIN;              //SD 卡关闭
       //send 8 Clock Impulse
       Write_Byte_SPI(0xFF);

       //set MMC_Chip_Select to low (MMC/SD-Card active)
      MMC_PORT&=~MMC_CS_PIN;              //SD 卡使能

       Write_Byte_SPI(cmd 0x40);       //送头命令
       Write_Byte_SPI(arg>>24);
       Write_Byte_SPI(arg>>16);          //send 6 Byte Command to MMC/SD-Card
       Write_Byte_SPI(arg>>8);
       Write_Byte_SPI(arg&0xff);
       Write_Byte_SPI(0x95);             //仅仅对 RESET 有效的 CRC 效验码
```

```
    //get 8 bit response
    //Read_Byte_MMC(); //read the first byte,ignore it.
    do
    {     //Only last 8 bit is used here.Read it out.
        tmp = Read_Byte_SPI();
        retry++;
    }
    while((tmp==0xff)&&(retry<100));     //当没有收到有效的命令的时候

    if(reading==0)
    MMC_PORT =MMC_CS_PIN;                    //MMC_CS_PIN=1;
    else MMC_PORT&=~MMC_CS_PIN;              //MMC_CS_PIN=0;
    return(tmp);
}
//*****************************************************************************
//SD 卡初始化(SPI-MODE)
//*****************************************************************************
uchar SD_Init(void)
{
    uchar retry,temp;
    uchar i;
    MMC_PORT&=~MMC_CS_PIN;              //SD 卡使能

   delay_nus(250);     //Wait MMC/SD ready...
    for (i=0;i<0x0f;i++)
    {
        Write_Byte_SPI(0xff); //send 74 clock at least!!!
    }
    //Send Command CMD0 to MMC/SD Card
    retry=0;

    do
    { //retry 200 times to send CMD0 command
      temp=SD_Write_Command(0,0);
      retry++;
      if(retry==100)
      {
       ;//CMD0 Error!
      }
    }
    while(temp!=1);

    //Send Command CMD1 to MMC/SD-Card
    retry=0;
```

```
    do
    { //retry 100 times to send CMD1 command
      temp=SD_Write_Command(1,0);
      retry++;
      if(retry==100)
      {
      ;
      }
    }
    while(temp!=0);
    retry=0;
     SD_Write_Command(16,512);          //设置一次读写 BLOCK 的长度为 512 个字节

    MMC_PORT =MMC_CS_PIN;       //MMC_CS_PIN=1;     //set MMC_Chip_Select to high
    return(0); //All commands have been taken.
}
//*****************************************************************************
//从 SD 卡读一个扇区      Return 0 if no Error.
//*****************************************************************************
uchar SD_Read_Block(unsigned long address)
{
    uchar temp=0;uint i=0;
    reading=1;
    temp=SD_Write_Command(17,address);          //读出 RESPONSE
    while (Read_Byte_SPI()!= 0xfe)
    {;}     //直到读取到了数据的开始头 0XFE,才继续
    for(i=0; i<512; i++)
{
BUFFER[i]=Read_Byte_SPI();
}
    Read_Byte_SPI();//CRC - Byte
    Read_Byte_SPI();//CRC - Byte
    reading=0;
    MMC_PORT =MMC_CS_PIN;               //关闭 SD 卡
    return(temp);
}
```